游易-程序主题分享合辑

专题 游易程序第125期 手游内存优化

目录

Tracemalloc遇见火焰图

张育铭 2019.10.04 09:18 809 10 1 查看原文

> 本文阐述了使用Tracemalloc和FlameGraph两个工具生成Python内存占用的火焰图的流程，笔者在使用过程中体验良好，推荐给各Python项目组使用，优化Python的内存占用，

火焰图是非常好用的一款观察代码热点的工具，其源码在GitHub上[1]，使用效果如图1所示，我们可以清晰地看出每个函数被调用耗的时间，然后快速分析和定位代码的性能问题。

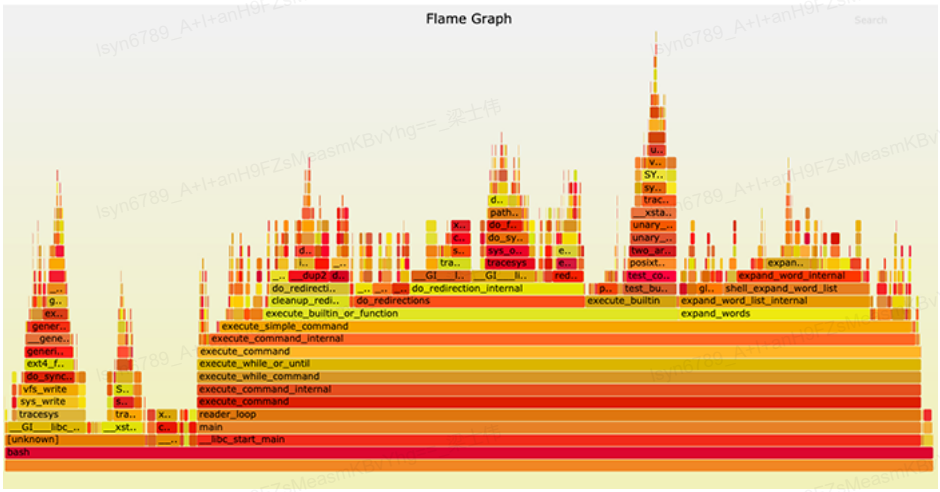

图1 火焰图示例

笔者所在的《一梦江湖》项目组是已经运营快两年的项目，脚本层代码非常庞大，Python脚本占用的内存大小不可小觑，笔者使用Tracemalloc[2]来分析脚本层的内存占用，Tracemalloc在Python模块启动的时候就开始运行，游戏启动后，可以通过tracemalloc._get_traces()这个函数，得到Python内存分配大小数据。

笔者使用Tracemalloc的数据分析内存占用的时候，发现Tracemalloc的输出非常不直观，需要进行多次整理和排序才能找到内存占用的热点，笔者将Tracemalloc输出的数据格式转化为火焰图（FlameGraph）所能识别的数据格式，将游戏的内存占用数据转化为火焰图，可以方便地观察内存热点，定位脚本层的内存问题，推荐给有需要优化Python内存的项目组使用。

下面介绍生成火焰图的步骤。

一、保存Tracemalloc数据

启动游戏后，执行图2所示的脚本，就可以得到Tracemalloc捕获的数据。

图2 获取Tracemalloc捕获的数据

图3显示了array中第0个元素的数据，其中208表示这次内存分配的数据大小为208 Bytes,'I:/H42/new_trunk/……/socket.py'表示分配的文件路径，47表示socket.py文件的47行执行的时候造成的内存分配。

```
>>>import tracemalloc
>>>array = tracemalloc._get_traces()
>>>array[0]
(208, (('I:/H42/new_trunk/code/src/TXM/Package/Script/Python/Lib\\socket.py', 47),), 'socket')
>>>array[0][1]
```

图3 Tracemalloc数据示例

二、格式化Tracemalloc数据

因为FlameGraph的输入数据格式是用分号分割的，因此我们执行图4所示的脚本，将Python的内存占用数据转化为分号分割的格

```
def gen_flamegraph_data(self):
    hexm_path_dict = {}
    for item in self._array:
        path = item[1][0][0]
        size = item[0]
        line_no = item[1][0][1]
        path_list = path.split("Python")
        if len(path_list) > 1 and path_list[1].endswith("py"):
            hexm_path = path_list[1]
            hexm_path_no = (hexm_path, line_no)
            if hexm_path_no not in hexm_path_dict:
                hexm_path_dict[hexm_path_no] = [size]
            else:
                hexm_path_dict[hexm_path_no].append(size)
    output_list = []
    for hexm_path_no, size_list in hexm_path_dict.iteritems():
        total_size = 0
        for size_item in size_list:
            total_size += size_item
        hexm_path_comma = ";".join(hexm_path_no[0].split("\\")[1:])
        line_no = hexm_path_no[1]
        output_list.append("%s;%s %s" % (hexm_path_comma, line_no, total_size))

    output_list.sort()
    final_str = "\n".join(output_list)
    with open("flamegraph_data.txt", "wb") as file_:
        file_.write(final_str)
        file_.write("\n")
```

图4 Tracemalloc数据转化为FlameGraph的输入数据格式

三、生成火焰图

笔者的FlameGraph环境是在Linux服务器上的，因此这一步我们将步骤二的flamegraph_data.txt数据传到Linux服务器上，使用[命令生成火焰图。

```
zymn4424@h42 ~/apps/FlameGraph]$ ./flamegraph.pl flamegraph_data.txt > flamegraph_data.svg
zymn4424@h42 ~/apps/FlameGraph]$
```

图5 使用flamegraph将数据转化为火焰图

生成的flamegraph_data.svg拖到浏览器里面如下图所示，可以清晰的看出hexm路径下的文件一共分配了76 Mbytes的内存，并件夹分配的内存大小非常清晰。

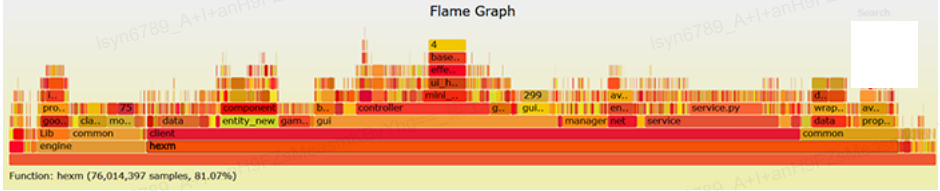

图6 《一梦江湖》Python内存分配火焰图示例

在浏览器中点击某个小段，支持放大显示，比如点击controller这一栏，得到图7所示的火焰图。

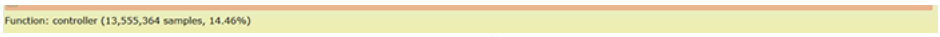

图7 《一梦江湖》controller目录分配火焰图示例

4 总结

通过前面的步骤我们可以顺利得到项目中Python内存分配的火焰图，作为程序同学就可以愉快的使用火焰图进行内存分析，优化的内存占用，减少玩家客户端的crash次数，提高玩家的用户体验。

1、https://github.com/brendangregg/FlameGraph

2、我们项目用的Python 2.7.x版本，把Python 3.x版本的tracemalloc移植了过来